

Resolution-Based Content Discovery in Network of Caches: Is the Control Traffic an Issue?

Bitá Azimdoost, Cedric Westphal, *Senior Member, IEEE*,
and Hamid R. Sadjadpour, *Senior Member, IEEE*,

Abstract—As networking attempts to cleanly separate the control plane and forwarding plane abstractions, it also defines a clear interface between these two layers. An underlying network state is represented as a view to act upon in the control plane. We are interested in studying some fundamental properties of this interface, both in a general framework, and in the specific case of content routing. We try to evaluate the traffic between the two planes based on allowing a minimum level of acceptable distortion in the network state representation in the control plane.

We apply our framework to content distribution, and see how we can compute the overhead of maintaining the location of content in the control plane. This is of importance to evaluate resolution-based content discovery in content-oriented network architectures: we identify scenarios where the cost of updating the control plane for content routing overwhelms the benefit of fetching the nearest copy. We also show how to minimize the cost of this overhead when associating costs to peering traffic and to internal traffic for network of caches.

I. INTRODUCTION

A communication network can be abstracted into two (logical) layers, namely, a control plane carrying signaling and administrative traffic, and a data forwarding plane carrying the user data traffic. In many applications, for the network to function properly, the control plane must have some knowledge about the forwarding plane in order to create a view of the underlying network. The underlying network will be in an operating state which is reported by a protocol to the control/management layer. For example, in a network of caches, the data plane contains caches keeping the data traffic, e.g. video or audio files, which are requested and used by the users, and the information regarding the items kept in each cache reported to the control layer forms the control traffic.

However, as the networks have grown in size and complexity, as end nodes, content and virtual machines move about, it will become more difficult for the control layer to have an accurate view of the forwarding plane. Consider the example of finding a service or a piece of content. Current protocols attempt to resolve a content request to the nearest copy of the object by using DNS or redirecting HTTP requests. Further proposals suggest to share content location information

in between content delivery networks (CDNs), or even to build content routing within the architecture. In all cases, this implicitly entails that the mechanism responsible to route to the content has to be dynamically updated with the content location. Meta-information from the forwarding plane needs to be delivered to the control plane. This raises the question: *how much?* In other words, depending on the size of the domain being controlled, of the underlying state space, of the dynamics of the evolution of the state in the forwarding plane, what stream of data is required to keep the control plane up to date?

We consider the issue of maintaining a consistent view of the underlying state at the control layer, and develop an abstracted mechanism, which can be applied to a wide range of scenarios. We assume the underlying state as an evolving random process, and calculate the rate that this process would create to keep the representation of this state up-to-date in the control plane. This provides a lower bound on the overhead bandwidth required for the control plane to have an accurate view of the forwarding plane¹.

We then illustrate the power of our model by focusing on the specific case of locating content in a resolution-based content-oriented network. Enabling content routing has attracted a lot of attention recently, and thus we are able to shed some light on its feasibility. In this case, the underlying state depends on the size and number of caches, on the request for content process and on the caching policy. We apply our framework to derive the bandwidth needed to accurately locate a specific piece of content. We observe that there is a trade-off for keeping an up-to-date view of the network at the cost of significant bandwidth utilization, versus the gain achieved by fetching the nearest copy of the content. We consider a simple scenario to illustrate this trade-off.

Our contribution is as follows:

- We present a framework to quantify the minimal amount of information required to keep a (logical) control plane aware of the state of the forwarding plane. We believe this framework to be useful in many distributed systems contexts.

B. Azimdoost and H. R. Sadjadpour are with the Department of Electrical Engineering, University of California, Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, USA (e-mail: {bazimdoost, hamid}@soe.ucsc.edu)

Bitá Azimdoost was with Huawei Innovation Center, Santa Clara, CA 95050, USA, as an intern while working on this paper.

Cedric Westphal is with the Department of Computer Engineering, University of California, Santa Cruz, 1156 High Street, Santa Cruz, CA 95064, USA and Huawei Innovation Center, Santa Clara, CA 95050, USA (e-mail: cedric.westphal@huawei.com)

¹There exists other overhead that we have not discussed in this work. We believe that addressing all the overhead of data/control plane interface in one paper may not be possible, since there might be several sources for them. However, if one thinks of certain sources of overhead, like the overhead of setting up a secure connection between the forward and control planes, then the actual protocol overhead would be proportional to the information theoretic overhead, at least if the rate of update is high enough. In which case we provide a good idea of how the whole protocol overhead will trend.

- We apply our framework to the specific case of locating content, and see how content location is affected by the availability of caches, the caching policy and the content popularity. We can thus apply our results to some of the content-oriented architectures and observe that cached copies would go ignored for a large swath of the content set.
- We see how our framework allows to define some optimal policies with respect to the contents that should be cached for an operator-driven content distribution system. While it is not surprising that very unpopular contents should not be cached, we can actually compute a penalty for doing so under our model.

We quickly note that our framework does not debate the merit of centralized vs distributed, as the control layer we consider could be either. For a routing example, our model would provide a lower-bound estimate of the bandwidth for, say OpenFlow to update a centralized SDN controller, or for a BGP-like mechanism to update distributed routing instances.

Our results are theoretic in nature, and provide a lower bound on the overhead. We hope they will provide a practical guideline for protocol designers to optimize the protocols which synchronize the network state and the control plane.

The rest of the paper is organized as follows. After going over some related work in section II, we introduce our framework to model the protocol overhead in section III, and then study the content location in the network of caches in section IV. The derived model is used to study a simple caching network as well. We show the power of the model in the protocol design by computing the cost of content routing in Section V and suggesting a cache management policy. Finally, section VI concludes the paper and describes some possible future work.

II. RELATED WORK

As SDN makes the separation explicit between the control and forwarding layers, it begs the question of how these layers interact. This interaction has been pointed out as one of the bottlenecks of OpenFlow [1], and several papers have been trying to optimize the performance of the traffic going from one layer to the other. For instance, [2] optimizes the controller to support more traffic, while [3] or [4] attempt to make the control layer more distributed and thus reduce the amount of interaction between the switches and the control layer. There has been no attempt to model the interaction between the control and forwarding layers to our knowledge.

Studying the gap between the state of the system and the view of the controller, [5] focuses on the relationship between performance and state consistency, and [6] studies similar relationship in multiple controller systems. This underlines the need for the view at the control layer to be representing the network state with as little distortion as possible.

The forwarding plane in a network usually consists of a state machine which is changing because of different network characteristics. The control plane needs to obtain adequate information about the underlying states so that the network can perform within a satisfactory range of distortion. The

first theoretical study of this information was conducted by Gallager in [7]. This work utilizes the rate distortion theory to calculate the bounds on the information required to show some characteristics such as the start time and the length of the messages.

The link states (validity of a link) and the geographic location and velocity of each node in a mobile wireless network are some examples of such state, which have been studied in [8] and [9], respectively. An information-theoretic framework to model the relationship between network information and network performance, and the minimum quantity of information required for a given network performance was derived in [10].

One impetus to study the relationship between the control layer and the network layer comes from the increased network state complexity from trying to route directly to content. Request-routing mechanisms have been in place for a while [11] and proposals [12] have been suggested to share information between different CDNs, in essence enabling the control planes of two domains to interact (our framework applies to this situation). And many architectures have been proposed that are oriented around content [13]–[18] and some have raised concerns about the scalability of properly identifying the location of up to 10^{15} pieces of content [19]. Our model presents a mathematical foundation to study the pros and cons of such architectures.

The cache management problem in the networks has been studied in several contexts. [20] presents a centralized approximation algorithm to solve the cache placement problem for minimizing the total data access cost in ad hoc networks. [21] proposes a replication algorithm that lets nodes autonomously decide on caching the information, and [22] determines whether/where to keep a copy of a content such that the overall cost of content delivery is minimized and show that such optimized content delivery significantly reduces the cost of content distribution and improves quality of service.

Some cooperative cache management algorithms are developed in [23] which tries to maximize the traffic volume served from cache and minimize the bandwidth cost in content distribution networks. [24] proposes some online cache management algorithms for Information Centric Networks (ICNs) where all the contents are available by caching in the network instead of a server or original publisher. [25] investigates if caching only in a subset of node(s) along the content delivery path in ICNs can achieve better performance in terms of cache and server hit rates. These works define a specific cost in the network and try to determine the locations and the number of copies of the contents in the network such that the defined cost is minimized. Finally [26] and [27] analytically prove that on-path content discovery has the same asymptotic capacity as finding the nearest copy in these networks.

To the best of our knowledge, there is no work considering the protocol overhead in such systems. In this work, we model the protocol overhead, then use that model to compute a general cost for data retrieval (including the protocol overhead). We also investigate whether allowing more copies of the contents cached in the network reduces the total cost. One related work on this topic is [28] which proposes a content

caching scheme, in which the number of chunks (fragments) to be cached in each storage is adjusted based on the popularity of the content. In this work, each upstream node recommends the number of chunks to be cached in the downstream node according to the number of requests.

III. PROTOCOL OVERHEAD MODEL

In this section we turn our attention to the mechanism to synchronize the view at the control layer with the underlying network state, and introduce a framework to quantify the minimal amount of required transferred information.

Assume that $S_X(t)$ describes the state of random process X in a network at time t . In order to update the control plane's information about the states of X in the network, the forwarding plane must send update packets regarding those states to the control plane whenever some change occurs. Let $\hat{S}_X(t)$ denote the control plane's perceived state of X at time t . It is obvious that no change in \hat{S}_X will happen before S_X changes, and if S_X changes, the control plane may or may not be notified of that change. Therefore, there are some instances of time where $\hat{S}_X \neq S_X$.

In this paper, we consider, systems and applications in which the state can have two values '0' and '1'.² For instance, a link can be up or down; or a piece of content can be present at a node, or not. Figure 1 illustrates the time diagram of state changes of such binary random process which is the state of the forwarding plane in the network being announced to the control plane.

Let $\{Y_m\}_{m=1}^{\infty}$ and $\{Z_m\}_{m=1}^{\infty}$ denote the sequences of '0's and '1's time durations of $S_X(t)$ respectively, and $\{T_m\}_{m=1}^{\infty}$ denote the times of changes. We consider large distributed systems, where the input is driven by a large population of users (smaller systems offer no difficulty in tracking in the control plane what is happening in the data plane). It is a well known result that the aggregated process resulting from a large population of uncoordinated users will converge to a Poisson process (chapter 3.6 [29]), and therefore the events in the future are independent of the events in the past and depend only on the current state. Thus we assume with no loss of generality that Y_m is an independent and identically distributed (i.i.d) sequence with probability density function (pdf) $f_Y(y)$ and mean θ_X , and Z_m is another i.i.d. sequence with pdf $f_Z(z)$ and mean τ_X . We also assume that any two Y_m and Z_m are mutually independent.

S_X and \hat{S}_X may differ in two cases resulting in two types of distortion; first, when the state of X is changed from '0'

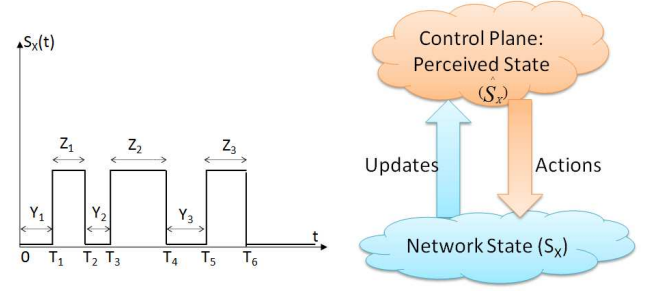


Fig. 1. Time diagram of the state of binary random process X at time t ($S_X(t)$).

to '1' (change type I) but the control plane is not notified ($\hat{S}_X = 0$, $S_X = 1$); second, when the state of X is changed from '1' to '0' (change type II) and the control plane still has the old information about it ($\hat{S}_X = 1$, $S_X = 0$). Let D_1 and D_2 denote the probability of the distortions corresponding to changes type I and II, respectively. Here we calculate the minimum rate at which the underlying plane has to update the state of X so that the mentioned distortion probabilities are less than some values ϵ_1 for the first type, and ϵ_2 for the second type, respectively. ϵ_1 and ϵ_2 can be viewed as probability of false negative and false positive alarms at the controller.

We make an additional assumption that the delay of the network is much lower than the time duration of the changes in the forwarding layer, and the control plane will be aware of the announced state immediately³ (the alternative - that the state of the system changes as fast or faster than the control plane can be notified of these changes - is obviously unmanageable). Thus, the above errors may occur just when the forwarding plane does not send an update about a change.

The main result now can be stated as a Lemma (with the proof in Appendix).

Lemma 1. If the ups and downs in the state of X follow some distributions with means τ_X and θ_X , respectively, then the minimum update rate $R_X(\epsilon_1, \epsilon_2)$ (number of update packets per second) satisfying the mentioned distortion criterion is given by

$$R_X(\epsilon_1, \epsilon_2) \geq \frac{1}{\tau_X + \theta_X} \left(2 - \frac{\epsilon_1 \frac{\theta_X}{\tau_X}}{\frac{\theta_X}{\tau_X + \theta_X} - \epsilon_2} - \frac{\epsilon_2 \frac{\tau_X}{\theta_X}}{\frac{\tau_X}{\tau_X + \theta_X} - \epsilon_1} \right) \quad (1)$$

²Note that this Boolean case is just an example to illustrate the method, and can be generalized to other possible values. For instance, to measure the congestion on a link, one could quantize the link congestion into bins (say bins b_1 to b_{10} for normalized link utilization between 0 to 0.1, 0.1 to 0.2, ..., 0.9 to 1) and map the link utilization to a 0/1 variable such that $b_h = 1$ if the current link utilization is in $((h-1)/10, h/10)$ and 0 otherwise. Obviously using this quantization method the b_h variables would not be independent and only one of them can be '1' at each instant of time. As other way to solve such problem, one can model the changes in the quantized levels as a binary variable. Since the values of the congestion levels change smoothly and there is not any kind of discontinuity in the congestion levels, one can expect going one level up or down in case of any changes. Using this method, one needs to have new distortion definitions. Due to the lack of space we leave it as future work to study other state distributions, where other distortion functions would apply.

³The control packets sent from the data plane to the control plane are very small in size comparing to the data packets. For example in case of transferring video files, no state changes happen in the data plane unless a video file is downloaded. Since the size of the video files are much larger than the update packets (hundreds of megabytes comparing to a few bytes), the download time and thus the duration of state changes is much lower than the delay of the network for update packets. According to [13] the request round-trip latency in Akamai and Cisco are in the order of a few 10ms, while the download time for a 1GB movie using a very high speed internet of 100Mbps would take around 10s. This is a practical assumption as well: if the delay in the network is longer than the duration of the changes, then a message sent to update the controller would be carrying obsolete information by the time it reaches the controller. Most practical systems are such that the time to notify the control is sufficient for the controller to use the information when it receives it. However, in any system, there is a chance that the actual state changes while the notification of the previous state is still underway, and there is always some distortion in the state representation at the control vs. the actual state in the forwarding plane.

if $\frac{\epsilon_2}{1-\epsilon_2} < \frac{\theta_X}{\tau_X} < \frac{1-\epsilon_1}{\epsilon_1}$ and $\epsilon_2\tau_X + \epsilon_1\theta_X < \frac{\tau_X\theta_X}{\tau_X+\theta_X}$. Otherwise the distortion criteria is satisfied with no update at all.

Lemma 1 shows the minimum update rate for state of a single random variable X in the underlying plane so that an accepted amount of distortion is satisfied. The total rate and consequently the total protocol overhead for keeping the control layer informed of the forwarding layer is the combination of all the overheads needed for all the random processes of the underlying layer, which may be independent of each other or have some mutual impacts.

In the following section, we will use our model to formulate the control traffic needed in the interaction between caches and controller inside a sub-network.

The notations used here can be found in Tables I-II.

IV. CONTENT LOCATION IN CACHE NETWORKS

Information-centric networks [30] usually employ routing-based [15] or resolution-based [14], [17], [31], [32] methods for content discovery purposes. In the routing-based discovery methods, like CCN, the required items are found exploring some areas of the network opportunistically or using other solutions like flooding. [33]–[35] have studied these methods and proposed solutions to have the best performance. Resolution-based methods, on the other hand, require the control layer to know at least one location for each piece of data. PSIRP, DONA, and NetInf (partly) are some models which use the resolution-based methods. For instance, [17] attempts to set up a route to a nearby copy by requesting the content from a pub/sub mechanism. The pub/sub rendezvous point needs to know the location of the content. This is highly dynamic, as content can be cached, or expunged from the cache at any time. NDN [16] also assumes that the routing plane is aware of multiple locations for a piece of content⁴.

In a cache network, the addition/removal of an item (pieces of data which are requested and used by the users) to/from a cache may affect the timings of the other items in that cache; caching a piece of content somewhere may force another content out of the cache, and the caching policy will thus influence the network state (the existing items information), so we need to consider this effect in our calculations. It is worth noting that this framework may be used for CDNs as well, since the basics are the same, the point is that the update traffic for reporting the state of the caches in CDN would be very close to zero, since there are not a large number of changes in their states, unless the acceptable distortion is very low. We assume from now on that the Least-Recently-Used (LRU) replacement policy is used in the caches, as it is a common policy and has been suggested in some ICN architectures [15]. However, based on [37], other policies can be handled in a similar manner by generalizing the decoupling technique of Che's approximation [38].

The request process also impacts the cache state, and we make the usual assumption that the items are requested

according to a Zipf distribution with parameter α ; meaning that the popularity of an item i is $\alpha_i = \frac{i^{-\alpha}}{\sum_{k=1}^M k^{-\alpha}}$, where M is the size of the content set.

In the following sections we first introduce our framework to model the protocol overhead in section III. Then, in section IV-A, we use our model to study the total data retrieval cost including the control overhead and data downloading costs in a network of caches, where the nodes update the control plane of a domain (say, an AS) so as to route content to a copy of the cache within this domain if it is available. We denote the control plane function which locates the content for each request as the Content Resolution System (CRS).

A. Cache-Controller Interaction

Assume that we partition the network into smaller sub-networks each with its own control plane, such that all the nodes in each one of them have similar request patterns. A possible example of such partitioning are the Autonomous Systems (ASs) in the Internet.

Whenever a client has a request for an item, it needs to discover a location of that item, preferably within the AS, and it downloads it from there. To do so, it asks a (logically) centralized *Content Resolution System (CRS)* by sending a Content Resolution Request (CRReq) or locates the content by any other non-centralized locating protocol. The Content Resolution Reply (CRRep) sent back to the client contains the location of the item, then the client starts downloading from the cache identified in CRRep.

If the network domain is equipped with a CRS, it is supposed to have the knowledge of all the caches, meaning that each cache sends its item states (local presence or absence of each item) to the CRS whenever some state changes.

Depending on the caching policy, whenever a piece of content is being downloaded, either no cache, all the intermediate caches on the path, or just the closest cache to the requester on the download path stores it in its content store, independently of the content state in the other caches, or refresh it if it already contains it.

We consider an autonomous network containing N nodes (terminals), each sending requests for items $i = 1, \dots, M$ with sizes B_i according to a Poisson distributed process with rate of γ_i . The total request rate for all the items from each node is denoted by $\gamma = \sum_{i=1}^M \gamma_i$. Note that all the nodes in an AS have the same request pattern, i.e. content locality is assumed uniform in each AS⁵, and that the total request rate of each terminal is a fixed rate independent of the total number of nodes and items while the total requests for all the nodes is a function of N (namely $N\gamma$). If different users have different

⁴The routing (in NDN in particular) could know only one route to the content publisher or to an origin server and find cached copies opportunistically on the path to this server. But Fayazbakhsh et. al. [36] have demonstrated that the performance of such an ICN architecture would bring little benefit over that of strict edge caching.

⁵This assumption is widely used in works using the mathematical modeling for the networks [39]. This comes from the fact that 1) The requests coming from a specific region are very likely to follow similar patterns, because the users' interest in one area are highly correlated and can be predicted by having the information about just part of them [40]. For example, some certain news title might be of special interest in a certain area, or some new TV series might be very popular in a certain country. 2) each user in this paper can actually be a hot spot or a base station, so a request generated from a node is not coming from one specific user but a group of users. So since we assume random users per station, then the assumption of uniform user locality is the best fitted assumption.

request distributions, then less cached contents will be reused, and thus there will be more changes in the cache states, and consequently more update traffic will be needed. The uniform content locality will give us the minimum required update rate.

Suppose that there are N_c caches in the system ($\mathcal{V}_c = \{v_1, \dots, v_{N_c}\}$) each with size L_c that can keep (and serve) any item i for some limited amount of time τ_i , which depends on the cache replacement policy. Based on the the rate at which each item i enters a cache and the time it stays there, each cache may have item i with some probability ρ_i . For simplicity, we assume that the probability distribution of the contents in all the caches are similar to each other. We can easily extend to the case of heterogeneous caches at the cost of notation complexity. For instance, Theorem 1 below can be stated as a sum over all N_c possible types of caches with N_c different ρ_i s for each type of cache, instead of a product by N_c of identical terms. Our purpose is to describe the homogeneous case, and let the reader adapt the heterogeneous case to suit her/his specific needs.

In the following Theorem we want to compute the update rate for this system. Let \bar{N}_c denote the number of caches where each downloaded piece of content is stored in (and thus need to send an update), either on the downloaded path, or off-path (Caching policy and network topology are the two factors that determine this number.). Thus, the rate of requests for item i received by each cache is $\lambda_i = \gamma_i N \bar{N}_c / N_c$.

Parameter	Definition
N	Number of users
M	Total number of items
N_c	Number of content stores/caches
\bar{N}_c	Number of caches storing the downloaded content
L_c	Storage size per content store (bits)
B_i	Average size of item i (bits)
α_i	Popularity of item i (Zipf)
γ_i	Total request rate for item i per user
γ	Total request rate per user
λ_i	Total request rate for item i received per cache
λ	Total request rate received per cache

TABLE I
Parameters of the network model

Theorem 1. *The minimum total update rate for each item i in the worst case is*

$$R_i(\epsilon_1, \epsilon_2) \geq N_c \lambda_i (1 - \rho_i) \left\{ 2 - \frac{\epsilon_1(1 - \rho_i)}{\rho_i(1 - \rho_i - \epsilon_2)} - \frac{\epsilon_2 \rho_i}{(1 - \rho_i)(\rho_i - \epsilon_1)} \right\} \quad (2)$$

if $\epsilon_1 < \rho_i < 1 - \epsilon_2$ and $\epsilon_1(1 - \rho_i) + \epsilon_2 \rho_i < \rho_i(1 - \rho_i)$. Otherwise no update is needed.

Proof: Let the random process X in the forwarding plane denote the existence of item i in a cache v_j at time t , which is needed to be reported to the control plane (CRS). Let τ_{ij} denote the mean duration of time item i spends in any cache v_j , and θ_{ij} denote the mean duration of time item i not being in the cache v_j .

In order to keep the CRS updated about the content states in the network, all the nodes have to send update packets regarding their changed items to CRS. All the assumptions of section III are valid here. Thus, by replacing τ_X and θ_X in equation (1) with τ_{ij} and θ_{ij} respectively, the result ($R_{ij} = R_X$) shows the minimum rate at which each cache v_j has to send information about item i to the CRS.

It can be seen that at the steady-state, the probability that cache v_j contains item i will be $\rho_{ij} = \frac{\tau_{ij}}{\theta_{ij} + \tau_{ij}}$. On the other hand, the total rate of generating (or refreshing) copies of item i at each cache v_j , denoted by λ_{ij} , equals to $\frac{1}{\theta_{ij}}$. Replacing the values of $\frac{\tau_{ij}}{\tau_{ij} + \theta_{ij}}$ and $\frac{1}{\theta_{ij}}$ in R_{ij} with ρ_{ij} and λ_{ij} respectively, we obtain

$$R_{ij}(\epsilon_1, \epsilon_2) \geq \lambda_{ij} (1 - \rho_{ij}) \left\{ 2 - \frac{\epsilon_1(1 - \rho_{ij})}{\rho_{ij}(1 - \rho_{ij} - \epsilon_2)} - \frac{\epsilon_2 \rho_{ij}}{(1 - \rho_{ij})(\rho_{ij} - \epsilon_1)} \right\} \quad (3)$$

for $\epsilon_1 < \rho_{ij} < 1 - \epsilon_2$ and $\epsilon_2 \rho_{ij} + \epsilon_1(1 - \rho_{ij}) < \rho_{ij}(1 - \rho_{ij})$.

It is worth noting that we are not assuming any specific topology or caching policy here; the items may be cached on-path or off-path; just one cache may keep the downloaded content or a few caches may keep it. We are looking for the minimum amount of update packets in the worst case, which happens when each cache stores items independent of the items in other caches. It is obvious that topologies like a line of caches which result in strongly dependent caches are not in the scope of this paper. Thus, the total update rate for item i , is the sum of the update rates in all caches which is $R_i(\epsilon_1, \epsilon_2) = \sum_{j=1}^{N_c} R_{ij}(\epsilon_1, \epsilon_2)$. Recalling the assumption of (probabilistically) similar caches, we can drop the index j and express the total update rate of item i in terms of the probability of this item being in a cache. This yields the result of equation (2) and the total update rate for all the items is the summation of these rates. ■

Parameter	Definition
τ_i	Average time item i stays in a cache
θ_i	Average time a cache does not have item i
ρ_i	Probability of item i being in a cache
$\epsilon_{1,2}$	Distortion thresholds
$R_{ij}(\epsilon_1, \epsilon_2)$	Minimum rate at which each cache v_j must send update state of item i to CRS so the defined distortion criteria is satisfied
$R_i(\epsilon_1, \epsilon_2)$	Minimum total update rate for item i that satisfies the defined distortion criteria
$R(\epsilon_1, \epsilon_2)$	Minimum total update rate that satisfies the defined distortion criteria

TABLE II
Parameters used in cache-controller interaction

B. Model Evaluation and Simulation Results

To figure out how the calculated rates perform in practice and evaluate our model, we simulate an LRU cache with capacity $L_c = 20$ items. We use the MovieLens dataset [41], which contains 100,000 ratings together with their time

stamps collected for $M = 1,682$ movies from 943 users during a seven-month period. We took the ratings as a proxy for content requests, assuming that the users who reviewed the movie have requested them shortly prior to the review. In these simulations we first estimate the item availability in the cache ρ_i (by dividing the total time that item is in the cache by the total simulation time), then using the estimated ρ_i and according to equations (17) and (18), we calculate the update rate in case of a change. Then we run the simulation for 100,000 requests. In this simulation we update the CRS according to the calculated rates, which can be interpreted as the chances of update, whenever a change occurs in the cache. Then we measure the total time that the CRS information does not match the actual cache state for each item, and calculate the average generated distortion during 10 rounds of simulation. The top figures in Figure 2 illustrate the results for the case where $\epsilon_1 = \epsilon_2 = 0.01$.

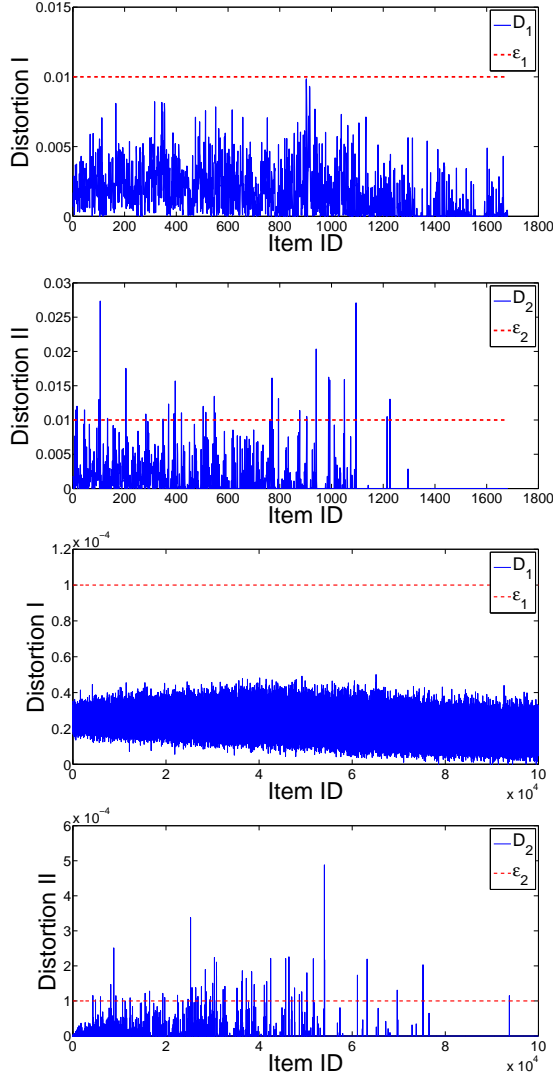


Fig. 2. Measured distortion type I (D_1) and II (D_2) top) for MovieLens dataset (100K requests for $M = 1,682$ movies), with $\epsilon_1 = \epsilon_2 = 0.01$ as accepted distortions, and bottom) for synthetic dataset (10M Poisson requests for $M = 100K$ Zipf distributed movies with skew factor $\alpha = 0.7$), with $\epsilon_1 = \epsilon_2 = 10^{-4}$ as accepted distortions.

Since, according to [14] and [42], the number of data objects is very large, and is becoming larger, we repeated similar evaluation with a relatively large synthetic dataset, containing 10 million Poisson requests for contents picked from a catalog of 100K movies, according to a Zipf distribution with skew parameter $\alpha = 0.7$. Bottom figures in Figure 2 show the results for the synthetic dataset allowing $\epsilon_1 = \epsilon_2 = 10^{-4}$ distortion accepted (larger number of contents leads to lower cache availability, thus we allowed lower distortion here).

It must be noted here that we are estimating ρ_i based on the past cache states, so it is not the exact ρ_i . Thus the generated distortion may exceed the tolerable values for some items, while they are in the safe zone for the others. It is observed that for a large portion of the items the distortion type I satisfies the distortion criteria. Distortion type II, however, does not satisfy the distortion criteria for more items. The reason is that the calculated update rates are strongly dependent on the availability of the items in the cache and any small error in the estimation of ρ_i may lead to some extra distortions. Since the ρ_i 's are mostly very small, not updating just one type II change may cause an error which remain in the system for a long time, and thus creating a large distortion.

Figure 3 illustrates the number of needed updates per generated request for each item i in the network $\frac{R_i}{N_c \lambda_i} = \frac{R_i}{N \gamma_i}$ when the caches does not contain it with a known probability $(1 - \rho_i)$. The only variable parameters in this graph are ϵ_1 and ϵ_2 . The higher distortion we tolerate, the less update announcements for each item i we need to handle. Moreover, the number of items which need some updates is decreasing when higher distortions are accepted. As can be seen the update rate starts from zero for those items which are in the cache with high probability. Status of these items are permanently set to '1' in CRS, and no update is needed. At the other end of the graph, for the items which are almost surely not in the cache, The presence probability is close to zero ($\rho_i = 0$ and thus $1 - \rho_i = 1$), and the status of those items can be permanently set to '0' in the control plane, thus the caches don't need to send any more information regarding those items to the control plane. Therefore, again no update is needed.

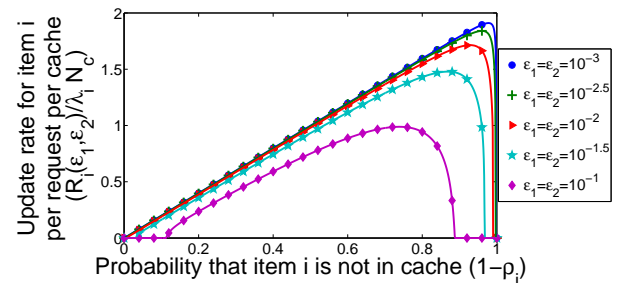


Fig. 3. Number of needed updates for item i sent from all caches to CRS per generated request for that item versus $1 - \rho_i$ for different distortion criteria.

The probability ρ_i is strongly dependent on the cache replacement policy. We consider LRU as the cache replacement method used in the network. Clearly, in LRU caches (similarly in other policies like FIFO, LFU, etc.) ρ_i is just a function of

the probability of item i coming to the cache (α_i), and the cache capacity (L_c). Figure 4 shows the changes of the total update rate (scaled by $\frac{1}{\lambda N_c}$) versus the cache storage size, in a network of LRU caches, such that the distortion criteria defined by (ϵ_1, ϵ_2) is satisfied. In this simulation $M = 10^3$. Note that each change in a cache consists of one item entering into and one other item being expunged from the cache, therefore if no distortion is tolerable, this rate will be 2 updates per change per cache.

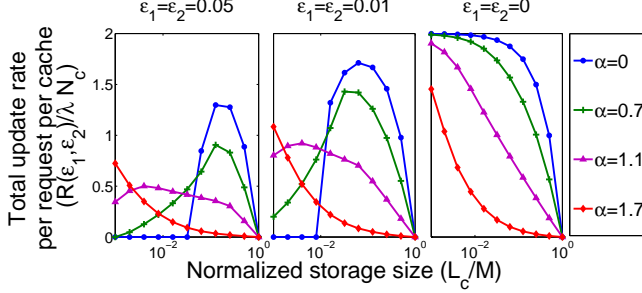


Fig. 4. Total cache-CRS update rate (Updates per generated request per cache) for different cache storage capacities and different acceptable distortions. Content set contains $M = 1,000$ contents.

It can be observed that for very small storage sizes and small popularity index, almost each incoming item changes the status of the cache and triggers an update. When the storage size is still very small, the caches do not provide enough space for storing the items and reusing them when needed, so increasing the size will increase the update rate. At some point, the items will move down and up in the cache before going out, so increasing the storage size more than that will reduce the need to update. However, if the popularity index is large, then increasing cache size from the very small sizes will decrease the need to update since there are just a few most popular items which are being requested.

Moreover, as it is expected, when more distortion is tolerable, the CRS needs fewer change notifications. However, if the cache size is too big, or the popularity exponent is too high, fewer changes will occur, but almost all the changes are needed to be announced to the CRS. On the other hand, for small cache sizes accepting a little distortion will significantly decrease the update rate.

V. APPLICATION TO COST ANALYSIS

In this section we use Theorem 1 to study trade-offs involved in updating the content control layer. More specifically, we try to calculate the bounds on the total cost (required bandwidth for download + CRS update) and look at the trade-offs between the cost, the size of the information chunks, the number of caches, and the size of caches.

A. Network Model

Figure 5 illustrates the network model studied in this section. This model consists of entities in three substrates: users are located in first layer; a network of caches with the CRS on the second level; external resources (caches in other networks, Internet, etc.) on the third.

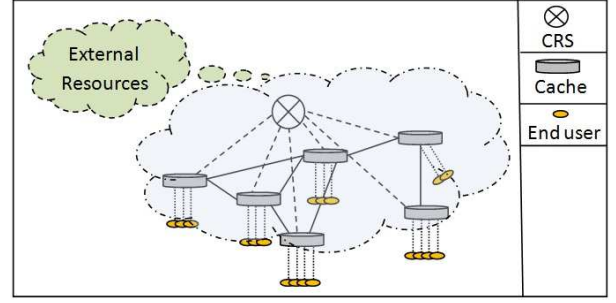


Fig. 5. Network Model.

We need to define the relative costs of the different actions. We assume that the state update process for item i has a per bit cost of ξ_i^{up} for sending data from the cache to CRS, and per bit cost of $\xi_i^{extup} > \xi_i^{up}$ for sending data from one CRS to another one. On the other hand, the requested piece of content i may be downloaded from a cache inside the network with some per bit cost ξ_i^{int} , or it may be downloaded from an external server with some other cost $\xi_i^{ext} > \xi_i^{int}$. These costs may be a function of the number of hops in the network. Note that the exact costs for each cache are determined based on the network topology, and may not be the same for all the caches. In this paper we use the average cost over all the caches ⁶.

B. Total Cost in Cache-Controller Interaction

Lemma 2. The length of each update packet for content i is

$$l_i \geq \log N_c - \log \frac{\lambda_i(1 - \rho_i)}{\sum_{k=1}^M \lambda_k(1 - \rho_k)} + 1 \quad (4)$$

Proof: Each update packet contains the ID of the cache issuing the query, the ID of the updated item and its new state. There are N_c caches in the network, hence, $\log N_c$ bits are needed to represent the cache. Item i is updated with probability $\beta_i = \frac{\lambda_i(1 - \rho_i)}{\sum_{k=1}^M \lambda_k(1 - \rho_k)}$, which results in a code length of at least $-\log \beta_i$ bits. Thus, the length of each update packet is $l_i \geq \log N_c - \log \beta_i + 1$. ■

Lemma 3. The total update cost in the defined network is

$$\varphi^{up} = \sum_{i=1}^M R_i(\epsilon_1, \epsilon_2) l_i \xi_i^{up}. \quad (5)$$

where $R_i(\epsilon_1, \epsilon_2)$ is the minimum rate at which the update state of item i must be reported to CRS so that a distortion criteria defined by (ϵ_1, ϵ_2) is satisfied.

Proof: Each cache sends update packets at rate $R_i(\epsilon_1, \epsilon_2)$ to provide its CRS with the state of item i in its local content store. Each update packet contains l_i bits, and there is a per bit cost of ξ_i^{up} for the update packets. Therefore, the cost

⁶The other option for defining the distortion and correspondingly cost is the worst case, which will map to the maximum update cost. There are many few caches that may undergo some maximum number of changes and need some maximum update transfers, thus, this is clearly not illustrating the performance of a cache network correctly. We have decided to work with the average method, which is very common in the literature (References [8]–[10]) and we believe it can represent the performance of the entire network better in our specific application and many others.

for updating information about item i in the sub-network is $\varphi_i^{up} = R_i(\epsilon_1, \epsilon_2)l_i\xi_i^{up}$, and the total update cost is $\varphi^{up} = \sum_{i=1}^M R_i(\epsilon_1, \epsilon_2)l_i\xi_i^{up}$. ■

Lemma 4. The total download cost in the defined network is

$$\varphi^{dl} = \sum_{i=1}^M N\gamma_i B_i((P_i - \rho_i)\xi_i^{int} + (1 - P_i)\xi_i^{ext}) \quad (6)$$

if $\epsilon_1 < \rho_i < 1 - \epsilon_2$ and $\epsilon_1(1 - \rho_i) + \epsilon_2\rho_i < \rho_i(1 - \rho_i)$. Otherwise no update is needed.

Proof: The requested piece of content i may be downloaded from the local cache with cost 0 (with probability ρ_i of being in this cache), from another cache inside the same network with some per bit cost ξ_i^{int} (with a probability we denote by $P_i - \rho_i$, where P_i is the probability that content i is within the AS's domain), or it must be downloaded from an external server with some other cost ξ_i^{ext} (with probability $(1 - P_i)$). Obviously, $\rho_i \leq P_i \leq 1$. Thus, the download cost for item i with size B_i bits for each user in the sub-network is

$$\varphi_{ij}^{dl} = \gamma_i B_i((P_i - \rho_i)\xi_i^{int} + (1 - P_i)\xi_i^{ext}), \quad (7)$$

The total download cost for item i is $\varphi_i^{dl} = N\varphi_{ij}^{dl}$, and the total download cost for all the items is the summation of φ_i^{dl} 's over all the contents. ■

Theorem 2. The total cost in the defined network including update and download costs is

$$\begin{aligned} \varphi &= \sum_{i=1}^M N\gamma_i B_i((P_i - \rho_i)\xi_i^{int} + (1 - P_i)\xi_i^{ext}) \\ &+ \sum_{i=1}^M R_i(\epsilon_1, \epsilon_2)l_i\xi_i^{up}. \end{aligned} \quad (8)$$

Proof: Adding Lemmas 3 and 4 proves the Theorem. ■

It can be seen that the total cost is strongly dependent on where each query is served from, and consequently on the probability of each item being internally served (P_i). This probability depends on the probability of that item being in an internal cache, which is in turn a function of the caching criteria and the replacement policy. Lemma 5 presents some bounds on P_i based upon the allowed distortion. The proof can be found in appendix.

Lemma 5. The probability that each content i is internally served is bounded by

$$[1 - (1 - \rho_i + \epsilon_1)^{N_c}]^+ \leq P_i \leq 1 - (1 - \rho_i)^{N_c}. \quad (9)$$

where $[x]^+ = \max(x, 0)$,

Note that the above P_i may take any value in the calculated bounds depending on the value of ρ_i . For example if $\rho_i < \epsilon_1$ then $D_{1_i} = \rho_i$, and $P_i = 0$, which is the lowest value of this bound. On the other hand, if $\rho_i > 1 - \epsilon_2$ then $D_{1_i} = 0$, and $P_i = 1 - (1 - \rho_i)^{N_c}$, which is the highest value in this bound. All the other values of ρ_i will lead to P_i between those two boundaries.

These two extreme cases of P_i result in some bounds on the download cost. Let φ_L^{dl} and φ_H^{dl} denote the lower and upper

bounds of the download cost corresponding to the upper and lower bounds of P_i , respectively, and φ_L and φ_H denote the lower and upper bounds of the total cost. Note that for small values of the tolerable distortion ϵ_1 the upper and lower limits of P_i and correspondingly the bounds of download cost are very close to each other.

Figure 6 the left plot illustrates the changes of update and download costs in a network with a content set of a total of 1 million contents, when the size of each cache is limited to $L_c = 100$ contents. The length of the data packets is assumed to be 100KB in average, while the update packets are l_i bytes each. Note that increasing the data (or update) packet lengths will increase the download (or update) cost linearly.

Here we assume that whenever an item is downloaded, it is stored in $\bar{N}_c = \log N_c$ caches, which have to report the changes to the controller⁷. If these caches are selected randomly, the total update rate would be \bar{N}_c times the rate of update of each cache resulting in $\max \varphi^{up}$. On the other side, if they are completely dependent, for example if all the caches on the download path keep it, then just one update may be enough, resulting in $\min \varphi^{up}$. So depending on the caching policy, the update cost will be something between \min and $\max \varphi^{up}$.

The request rate received by each cache is inversely proportional to the number of caches (the request rate per user is assumed to be fixed and independent of N_c), and the update packet length increases logarithmically with the number of caches. The total update rate per cache is almost linearly decreasing with N_c , hence the minimum of the total update rate, or the total update rate if just one cache keeps the downloaded item, will almost be stable when N_c varies (changes are in the order of $\log N_c$). The maximum total update rate will linearly change with the number of copies \bar{N}_c per download. Thus, increasing the number of caches in the network increases the update cost by a factor of at least $\log N_c$ and at most $\bar{N}_c \log N_c$.

Increasing N_c , however, increases the probability of an item being served internally and thus decreases the download cost. Nevertheless, as it can be observed, the rate of decrease is so low that it can be assumed as stable.

In the right plot of figure 6, we fix the number of caches in the AS ($N_c = 10$) and study the effects of cache storage size on the update and total cost. Increasing the cache size, simply increases the probability of an item being served internally and decreases the download cost. Again similar to the left plot, the rate of changes in the download cost is very low. As expected, on the other side, the update cost shows an increase when increasing the storage size. Looking at each cache, very small cache size leads to very large durations where that item is not in that cache and consequently, the update rate would be low. Increasing the storage size will increase the probability of that item being in the cache, and thus increases the update rate. If we let more cache storage, this increase reaches its highest value for a certain value of cache size, and for larger values of cache beyond a threshold, the item is in the cache most of

⁷This happens in largely used network models like binary tree or grid topology, when all the caches on the download path store the content.

the time. Therefore, we need less updates and increasing the cache size will increase the duration of the item being in the cache leading to fewer update messages. Since the total cost mostly depends on the download cost, by increasing the cache size, this value reaches its minimum value.

It is worth noting here that the cost of download from another AS has been assumed 5 times bigger than the cost of download from inside the AS, which in turn is assumed to be the same as the update cost per bit. Figure 7 shows the impact of the external and internal costs on the total download cost. Higher external costs result in higher total download costs, as it is expected, and show more decrease rate when the number of caches increases. Thus, if the external download cost comparing to the internal download cost is very high, having more caches may make sense, although, the total cost decrease rate is still very low.

Another important result shown by figures 6 and 7 is that having big data packets, the download cost is always much higher than the update cost, which is reasonable. In other words, having the resolution-based content discovery when the data packets are large, will add just negligible cost to update the controller. In the following we study the affect of chunk-based caching to obtain some insight on the reasonable size of chunks, such that the update cost remains negligible.

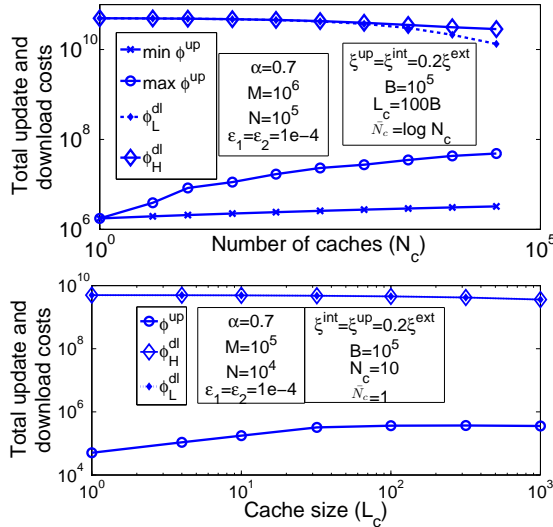


Fig. 6. Total update cost (minimum and maximum of φ^{up}) and total download cost (lower and upper bounds, φ_L^{dl} and φ_H^{dl} , left) vs. the number of caches (N_c), when each item is $B = 10^5$ units long, the storage size per cache is fixed ($L_c = 100$ items), and each downloaded item is stored in $\bar{N}_c = \log N_c$ caches, and right) vs. the cache size (L_c), when each item is $B = 10^5$ units long, the number of caches is fixed ($N_c = 10$), and each downloaded item is stored in $\bar{N}_c = 1$ cache.

The top plot in figure 8 shows the total cost versus the number of caches, when the LRU cache replacement policy is used and the total storage of the cache sublayer is limited to half of the total number of items. The parameters are set as follows: $M = 10,000$, $\epsilon_{1,2} = 1e-4$, $B_i = 100K$, $\xi_i^{int} = 1$, $\xi_i^{ext} = 5$, $\xi_i^{up} = 1$, and $\alpha = 0.7$. Since the lower and upper bounds of the total cost are very close to each other, we just plot the upper limit here. In the bottom plot of this figure, the total cost is plotted versus the size of each cache. It can be

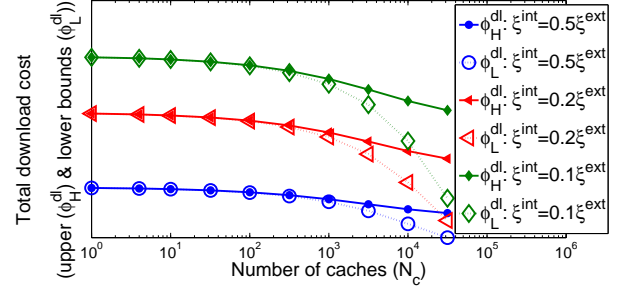


Fig. 7. Total download cost (lower and upper bounds, φ_L^{dl} and φ_H^{dl}) vs. the number of caches (N_c), for different download cost values, when each item is $B = 10^5$ units long, and the storage size per cache is fixed ($L_c = 100$ items).

observed that with a fixed total storage size, concentrating all the caches in one node and increasing the size of it will lead to better overall performance (least cost). Note that in these figures the total cost value shown is just a relative value, and not the exact one.

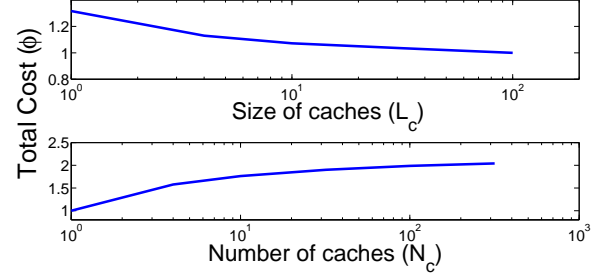


Fig. 8. Total cost (φ), when the total storage size ($N_c L_c$) is fixed and equal to half of the catalog size, vs. top) the size of caches (L_c), and bottom) the number of caches (N_c).

C. Optimized Cache Management

In previous section, the total cost in the described cache network was derived and the impacts of the number or size of the content stores on this cost was studied. We now turn our attention to minimizing the total cost for given N_c and L_c .

Under a Zipf popularity distribution, many rare items will not be requested again while they are in the cache under the LRU policy. We can rewrite the total cost if the caches only keep the items with popularity from 1 up to i^* .

$$\begin{aligned} \varphi = & \sum_{i=1}^{i^*} N \gamma_i B_i ((P_i - \rho_i) \xi_i^{int} + (1 - P_i) \xi_i^{ext}) \\ & + \sum_{i=i^*+1}^M N \gamma_i B_i \xi_i^{ext} + \sum_{i=1}^{i^*} R_i (\epsilon_1, \epsilon_2) l_i \xi_i^{up} \end{aligned} \quad (10)$$

Now just i^* different pieces of content may be stored in each cache. This changes the probability of an item $i = 1, \dots, i^*$ being in a cache (ρ_i), which in turn changes P_i and R_i .

Figures 9 demonstrates the total cost versus the caching popularity threshold i^* , for different number and size of content stores, and acceptable distortions.

If just a very small number of items (small i^*) are kept inside cache layer, then the download cost for those which are not allowed to be inside caches will be the dominant factor in the total cost and will increase it. On the other hand, if a lot of popularity classes are allowed to be kept internally, then the update rate is increased and also the probability of the most popular items being served internally decreases, so the total cost will increase. There is some optimum caching popularity threshold where the total cost is minimized. This optimum threshold is a function of the number and size of the stores, distortion criteria, per bit cost of downloads and updates.

The benefit of the optimized solution also varies depending on the mentioned parameters. For example according to figure 9, the optimized solution can have 17% reduction in cost in case when $N_c = 50, L_c = 10, \epsilon_1 = \epsilon_2 = 1e-4$, while this cost reduction is just 7% when N_c is five times smaller.

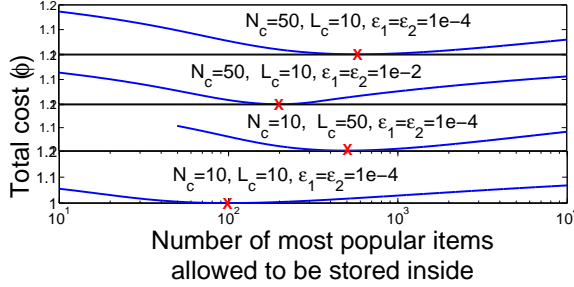


Fig. 9. Total cost when just i^* most popular items are allowed to be stored inside caches ($N = 10^3, M = 10^4, B = 10^5, \alpha = 0.7$).

To find the optimal i^* , assume that all the items have the same size ($B_i = B$) and the per bit costs is fixed for all popularity classes ($\xi_i^{int} = \xi^{int}, \xi_i^{ext} = \xi^{ext}, \xi_i^{up} = \xi^{up}$). We can rearrange equation (10).

$$\varphi = \varphi_1 - \varphi_2 + \varphi_3 \quad (11)$$

where $\varphi_1 = BN\gamma\xi^{ext}$ is the total cost if no cache exists and all the requests are served externally; $\varphi_2 = BN\gamma(\xi^{ext} - \xi^{int}) \sum_{i=1}^{i^*} \alpha_i P_i + BN\gamma\xi^{int} \sum_{i=1}^{i^*} \alpha_i \rho_i$ corresponds to the benefit of caching (cost reduction due to caching); and $\varphi_3 = \xi^{up} \sum_{i=1}^{i^*} R_i(\epsilon_1, \epsilon_2) l_i$ is the caching overhead cost due to the updates. The last term is the cost we pay because of caching (updates). We need to calculate the value of i^* such that the cost of caching is dominated by its advantage; i.e. we need to maximize $\varphi_2 - \varphi_3$.

This can be done using numerical methods which will lead to a unique i^* for each network setup (fixed parameters). However, the network characteristics and the request pattern are changing over time, so it seems that it is better to have a mechanism to dynamically optimize the cost by selecting the caching threshold (i^*) according to the varying network features.

In such a mechanism, the CRS can keep track of requests and have an estimation of their popularity. For those requests which are served locally the CRS can have an idea of the popularity based on the updates that receives from all the caches; i.e. the longer an item stays in a cache, the more

popular it is. It can also take into account the local popularity of the items. The CRS can then dynamically search for the caching threshold which minimizes the total cost by solving equation (11). Once the CRS determines which items to keep internally, it will set/reset a flag in each CRRep so that the local cache knows to store or not to store the requested piece of content.

VI. CONCLUSION AND FUTURE WORK

We formulated a distortion-based protocol overhead model. Some simple content distribution networks were then considered as examples to show how this framework can be used, and based on this model the overhead of keeping the control plane informed about the states of the contents in these networks was calculated. It was confirmed that with big data packets, or in large un-chunked data transfer scenarios, the cost of updating the control layer is much lower than the cost of data download, so resolution-based content discovery can be a good solution.

We also studied the total cost of data retrieval and observed that with limited cache storage sizes, allowing all the items to have the opportunity to be stored inside the sub-network's caches is not always the most efficient way of using the caching feature.

For the case with a central resolution system in each sub-network and with LRU cache replacement policy, an algorithm has been proposed that can dynamically determine which items not to be cached inside the AS at any time such that the total cost of data retrieval is minimized.

In this work, our overhead model focuses on systems with Boolean states. Our future work involves systems with other state distributions. In addition, we have assumed uniform distribution of caches in the studied example. This assumption means that the probability of an item being in all the caches are the same. Future study can consider some structure like tree or power-law for the caches inside each sub-network, and using the described framework, investigate how this assumption changes the results.

APPENDIX

Proof of Lemma 1

Proof: The distortion criteria is defined as

$$\begin{aligned} D_1 &= Pr(S_X = 1, \hat{S}_X = 0) \leq \epsilon_1 \\ D_2 &= Pr(S_X = 0, \hat{S}_X = 1) \leq \epsilon_2 \end{aligned} \quad (12)$$

It can be seen that $Pr(S_X = 1) = \frac{\tau_X}{\tau_X + \theta_X}$, and $Pr(S_X = 0) = \frac{\theta_X}{\tau_X + \theta_X}$. There are three cases where the distortion criteria is satisfied even when the controller has no information about the underlying plane.

- 1) If the monitoring state is 'down' with high probability ($Pr(S_X = 1) \leq \epsilon_1$), then having the controller assume that it is always 'down' (keeping \hat{S}_X constantly equal to '0') will satisfy the distortion criteria ($D_1 = Pr(S_X = 1) \leq \epsilon_1$ and $D_2 = 0 < \epsilon_2$).
- 2) If the monitoring state is 'up' with high probability ($Pr(S_X = 0) \leq \epsilon_2$), then setting the controller to assume it is always 'up' (keeping \hat{S}_X constantly equal

to '1') will satisfy the distortion criteria ($D_1 = 0 < \epsilon_1$ and $D_2 = Pr(S_X = 0) \leq \epsilon_2$).

- 3) If the monitoring variable can take both 'up' and 'down' states with high enough probabilities such that $1 - \frac{\epsilon_1}{Pr(S_X=1)} \leq \frac{\epsilon_2}{Pr(S_X=0)}$, then we pick a value ρ_0 between $1 - \frac{\epsilon_1}{Pr(S_X=1)}$ and $\frac{\epsilon_2}{Pr(S_X=0)}$, and assign '1' to \hat{S}_X with probability ρ_0 independent of the value of S_X . Therefore, since $D_1 = Pr(S_X = 1)Pr(\hat{S}_X = 0) = Pr(S_X = 1)(1 - \rho_0) \leq \epsilon_1$, and $D_2 = Pr(S_X = 0)Pr(\hat{S}_X = 1) = \rho_0 Pr(S_X = 0) \leq \epsilon_2$, the distortion criteria is satisfied.

Thus in the following, we concentrate on the cases where $Pr(S_X = 1) > \epsilon_1$, $Pr(S_X = 0) > \epsilon_2$, and $1 - \frac{\epsilon_1}{Pr(S_X=1)} > \frac{\epsilon_2}{Pr(S_X=0)}$.

Note that we assume that $\epsilon_1 + \epsilon_2 \leq 1$, then $\frac{\epsilon_2}{1-\epsilon_2} \leq \frac{1-\epsilon_1}{\epsilon_1}$, and the first two regions can be summarized in the region where $\frac{\epsilon_2}{1-\epsilon_2} \leq \frac{\theta_X}{\tau_X} \leq \frac{1-\epsilon_1}{\epsilon_1}$. The third region is also mapped to the region where $\epsilon_2 \tau_X + \epsilon_1 \theta_X < \frac{\tau_X \theta_X}{\tau_X + \theta_X}$.

Let $U_X^1(\epsilon_1)$ (and $U_X^2(\epsilon_2)$) denote the needed update rate per change type I (and II), or in other words the ratio of times that type I (and II) changes have to be reported to the control plane so that the distortion criteria is satisfied. As can be seen in figure 1, each 'up' period Z_m starts at time T_{2m-1} and ends at time T_{2m} . The false negative alarm is generated during the m^{th} 'up' period (Z_m) if a type I change in the state of X at time T_{2m-1} is not announced to the control plane while the previous state ('0') was correctly perceived by the control plane; we show this event by W_m^1 , and its probability is given by

$$\begin{aligned} Pr(W_m^1) &= (1 - U_X^1(\epsilon_1))Pr(\hat{S}_X = 0|S_X = 0) \\ &= (1 - U_X^1(\epsilon_1))(1 - Pr(\hat{S}_X = 1|S_X = 0)) \\ &= (1 - U_X^1(\epsilon_1))(1 - \frac{Pr(S_X=0, \hat{S}_X=1)}{Pr(S_X=0)}) \\ &= (1 - U_X^1(\epsilon_1))(1 - D_2 \frac{\tau_X + \theta_X}{\theta_X}) \end{aligned} \quad (13)$$

In this case, $\hat{S}_X = 0$ during the time where $S_X = 1$. So assuming that the m^{th} such change is perceived wrong by the control plane, Z_m is the time interval where the control plane has the type I wrong information about the state of X . Let N_w be the number of times S_X undergoes type I changes during a time interval $[0, w]$. The probability of type I error, and consequently type I distortion can be calculated as the ratio of total time of type I error over w when $w \rightarrow \infty$.

$$\begin{aligned} D_1 &= E[\frac{1}{w} \sum_{m=1}^{N_w} 1_{[W_m^1]} Z_m] \\ &= \frac{1}{w} E[1_{[W_m^1]} Z_m] E[N_w] \\ &= \frac{\tau_X}{\tau_X + \theta_X} Pr(W_m^1) \\ &= \frac{\tau_X}{\tau_X + \theta_X} (1 - U_X^1(\epsilon_1))(1 - D_2 \frac{\tau_X + \theta_X}{\theta_X}) \end{aligned} \quad (14)$$

Similarly, a false positive alarm is generated when a type II change is not announced while the previous perceived state ('1') was correct, and assuming that this is the m^{th} such change, Y_{m+1} is the time interval that the control plane has

type II wrong information about X ; let W_m^2 denote this event. Thus,

$$\begin{aligned} Pr(W_m^2) &= (1 - U_X^2(\epsilon_2))Pr(\hat{S}_X = 1|S_X = 1) \\ &= (1 - U_X^2(\epsilon_2)) \frac{Pr(S_X=1) - Pr(S_X=1, \hat{S}_X=0)}{Pr(S_X=1)} \\ &= (1 - U_X^2(\epsilon_2))(1 - D_1 \frac{\tau_X + \theta_X}{\tau_X}) \end{aligned} \quad (15)$$

and

$$\begin{aligned} D_2 &= E[\frac{1}{w} \sum_{m=1}^{N_w} 1_{[W_m^2]} Z_m] \\ &= \frac{1}{w} E[1_{[W_m^2]} Y_{m+1}] E[N_w] \\ &= \frac{\theta_X}{\tau_X + \theta_X} Pr(W_m^2) \\ &= \frac{\theta_X}{\tau_X + \theta_X} (1 - U_X^2(\epsilon_2))(1 - D_1 \frac{\tau_X + \theta_X}{\tau_X}) \end{aligned} \quad (16)$$

To satisfy the distortion criteria we need $D_1 \leq \epsilon_1$ and $D_2 \leq \epsilon_2$. The update rates per changes type I and II, $U_X^1(\epsilon_1)$ and $U_X^2(\epsilon_2)$, then can be written as

$$U_X^1(\epsilon_1) = 1 - \frac{D_1 \frac{\theta_X}{\tau_X}}{\frac{\theta_X}{\tau_X + \theta_X} - D_2} \geq 1 - \frac{\epsilon_1 \frac{\theta_X}{\tau_X}}{\frac{\theta_X}{\tau_X + \theta_X} - \epsilon_2} \quad (17)$$

$$U_X^2(\epsilon_2) = 1 - \frac{D_2 \frac{\tau_X}{\theta_X}}{\frac{\tau_X}{\tau_X + \theta_X} - D_1} \geq 1 - \frac{\epsilon_2 \frac{\tau_X}{\theta_X}}{\frac{\tau_X}{\tau_X + \theta_X} - \epsilon_1} \quad (18)$$

It can easily be verified that using the lower bounds obtained in equations (17) and (18) for update rates per each change type will result in distortions $D_1 = \epsilon_1$ and $D_2 = \epsilon_2$, and thus they are the minimum values needed.

Therefore, the total number of updates announced to the control plane divided by the total number of changes is given by

$$U_X(\epsilon_1, \epsilon_2) = U_X^1(\epsilon_1) + U_X^2(\epsilon_2) \quad (19)$$

Note that the total rate of type I changes, which is equal to the rate of type II changes in average is given by $\frac{1}{\tau_X + \theta_X}$ changes per second, thus total number of updates per second is given by

$$R_X(\epsilon_1, \epsilon_2) = \frac{U_X(\epsilon_1, \epsilon_2)}{\tau_X + \theta_X} \quad (20)$$

Combining equations (17-20) proves the Lemma. ■

Proof of Lemma 5

Proof: Recall that \mathcal{V}_c is the set of caches, ρ_i denotes the probability that a specific cache contains item i . Let S_{ij} represent the state of an item i at a node j , which is 1 if cache j contains item i , and 0 otherwise, and let \hat{S}_{ij} denote the corresponding state perceived by the CRS. A request from a user is not served internally (by a cache in second layer) either if no cache contains it:

$$Pr(\forall j \in \mathcal{V}_c : S_{ij} = 0) = (1 - \rho_i)^{N_c}, \quad (21)$$

or if there are some caches containing it but the CRS is not aware of that:

$$\begin{aligned}
& Pr(\exists j \in \mathcal{V}_c : S_{ij} = 1 \ \& \ \hat{S}_{ij} = 0) \\
&= \sum_{k=1}^{N_c} \sum_{1 \leq j_1 < \dots < j_k \leq N_c} Pr(\{i \notin \mathcal{V}_c - \{j_1, \dots, j_k\} \ \& \ \{\hat{S}_{ij_l}=0, S_{ij_l}=1\}_{l=1}^k\}) \\
&= \sum_{k=1}^{N_c} \sum_{1 \leq j_1 < \dots < j_k \leq N_c} (1 - \rho_i)^{N_c - k} \prod_{l=1}^k Pr(\hat{S}_{ij_l}=0) \\
&= \sum_{k=1}^{N_c} \binom{N_c}{k} (1 - \rho_i)^{N_c - k} D_{1_i}^k \\
&= (1 - \rho_i + D_{1_i})^{N_c} - (1 - \rho_i)^{N_c} \tag{22}
\end{aligned}$$

where $D_{1_i} \geq 0$ is the probability that i exists in cache j and the CRS does not know about it.

Thus the probability that a request is served externally is $1 - P_i$ which equals

$$\begin{aligned}
& (1 - \rho_i)^{N_c} + [(1 - \rho_i + D_{1_i})^{N_c} - (1 - \rho_i)^{N_c}] \\
&= (1 - \rho_i + D_{1_i})^{N_c} \tag{23}
\end{aligned}$$

where under the independent cache assumption, the state of an item in a cache is independent of the state in another cache. The probability $D_{1_i} \geq 0$ is always less than the probability of i being in cache j ($D_{1_i} \leq \rho_i$), and if the state updates are done at rate greater than $R_i(\epsilon_1, \epsilon_2)$, it will also be less than ϵ_1 . ■

REFERENCES

- [1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, Mar. 2008.
- [2] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," *Hot-ICE*, vol. 12, pp. 1–6, 2012.
- [3] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," *ACM SIGCOMM CCR*, vol. 41, no. 4, Aug. 2011.
- [4] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," *ACM SIGCOMM CCR*, vol. 41, no. 4, Aug. 2010.
- [5] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *ACM HotSDN*, Aug. 2012.
- [6] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *IEEE CNSM*, 2013, pp. 18–25.
- [7] R. Gallager, "Basic limits on protocol information in data communication networks," *IEEE Transactions on Information Theory*, vol. 22, no. 4, pp. 385–398, 1976.
- [8] D. Wang and A. Abouzeid, "Link State Routing Overhead in Mobile Ad Hoc Networks: A Rate-Distortion Formulation," in *IEEE INFOCOM*, Apr. 2008, pp. 1337–1345.
- [9] —, "On the cost of knowledge of mobility in dynamic networks: An information-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 11, no. 6, pp. 995–1006, Jun. 2012.
- [10] J. Hong and V. O. Li, "Impact of information on network performance-an information-theoretic perspective," in *IEEE GLOBECOM*, 2009, pp. 1–6.
- [11] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Known content network (CN) request-routing mechanisms," IETF RFC 3568, Network Working Group, Jul. 2003.
- [12] L. Peterson and B. Davie, "Framework for CDN interconnection," IETF CDNi working group, draft-ietf-cdni-framework-02, Dec. 2012.
- [13] M. Gritter and D. R. Cheriton, "An architecture for content routing support in the internet," in *USITS*, vol. 1, 2001, pp. 4–4.
- [14] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM CCR*, vol. 37, no. 4, 2007, pp. 181–192.
- [15] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *ACM CoNEXT*, 2009, pp. 1–12.
- [16] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project." Citeseer, 2010.
- [17] D. Trossen, G. Parisi, K. Visala, B. Gajic, J. Riihijarvi, P. Flegkas, P. Sarolahti, P. Jokela, X. Vasilakos, C. Tsilopoulos, and S. Arinifar, "Pursuit conceptual architecture: Principles, patterns and sub-components descriptions," Tech. Rep., May 2011.
- [18] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, *IEEE Communications Magazine*, no. 7, pp. 26–36, Jul.
- [19] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *ACM HotNets-X*, Nov. 2011.
- [20] B. Tang, H. Gupta, and S. R. Das, "Benefit-based data caching in ad hoc networks," *IEEE transactions on Mobile Computing*, vol. 7, no. 3, pp. 289–304, 2008.
- [21] S. Bhattacherjee, K. Calvert, and E. Zegura, "Self-organizing wide-area network caches," in *IEEE INFOCOM*, 1998.
- [22] B. Azimdoost, G. Farhadi, N. Abani, and A. Ito, "Optimal in-network cache allocation and content placement," in *IEEE INFOCOM WKSHPs*, 2015, pp. 263–268.
- [23] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *IEEE INFOCOM*, Mar. 2010.
- [24] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas, "Autonomic cache management in Information-Centric Networks," in *IEEE NOMS*, Apr. 2012.
- [25] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *International Conference on Research in Networking*. Springer, 2012, pp. 27–40.
- [26] B. Azimdoost, C. Westphal, and H. R. Sadjadpour, "On the throughput capacity of information-centric networks," in *IEEE ITC25*, 2013, pp. 1–9.
- [27] —, "Fundamental limits on throughput capacity in information-centric networks," *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 5037–5049, 2016.
- [28] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *IEEE INFOCOM WKSHPs*, 2012, pp. 316–321.
- [29] R. Durrett, *Probability: theory and examples*. Cambridge university press, 2010.
- [30] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014.
- [31] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, and H. Karl, "Network of information (netinf)—an information-centric networking architecture," *Computer Communications*, vol. 36, no. 7, pp. 721–735, 2013.
- [32] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "Mdht: a hierarchical name resolution service for information-centric networks," in *ACM SIGCOMM ICN*, 2011, pp. 7–12.
- [33] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiseelan, and J. Crowcroft, "Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking," in *ACM ICN*, 2015, pp. 9–18.
- [34] M. Lee, J. Song, K. Cho, S. Pack, J. Kangasharju, Y. Choi *et al.*, "Content discovery for information-centric networking," *Computer Networks*, vol. 83, pp. 1–14, 2015.
- [35] C. Anastasiades, A. Uruqi, and T. Braun, "Content discovery in opportunistic content-centric networks," in *IEEE LCN WKSHPs*, 2012, pp. 1044–1052.
- [36] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," in *ACM SIGCOMM CCR*, vol. 43, no. 4, 2013, pp. 147–158.
- [37] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *IEEE INFOCOM*, Apr. 2014, pp. 2040–2048.
- [38] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, 2002.

- [39] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE INFOCOM*, vol. 1, 1999, pp. 126–134.
- [40] E. Baştuğ, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *Journal of Communications and Networks*, vol. 17, no. 6, pp. 549–557, 2015.
- [41] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, p. 19, 2016.
- [42] J. Bankoski, *IEEE ICNC'17 Panel on Future Video Distribution*, 2017.